IMPLEMENTACIÓN DEL MÉTODO DE RECONOCIMIENTO DE OBJETOS EN IMÁGENES

Presentado por:

DANIEL JOSE GIRALDO TOBON

JUAN PABLO CORTES GONZALEZ

LAURA ALZATE MADRID

Materia

ANÁLISIS NUMÉRICO

Profesor

ALEJANDRO ARENAS VASCO

INGENIERÍA DE SISTEMAS

DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS

ESCUELA DE INGENIERÍA

UNIVERSIDAD EAFIT

MEDELLÍN

2021

Tabla de contenidos

1. Resumen

La importancia del reconocimiento de objetos o personas dentro de imágenes se ha convertido en algo muy significativo, ya que se ha usado para atrapar el rostro de personas buscadas o con procesos fiscales en distintos países como por ejemplo China. También ayuda en los procesos estadísticos. Uno de los ejemplos más cercanos a nosotros serían las cámaras de foto detección.

Abstract

The importance of the recognition of objects or people within images has become very significant, since it has been used to catch the faces of wanted people or with tax processes in different countries such as China. It also helps in statistical processes. One of the examples closest to us is the photodetection camera.

2. Introducción

La presente investigación está elaborada con el fin de encontrar el método más eficiente y a la vez menos complejo para la detección de objetos dentro de imágenes, utilizando diferentes técnicas y procedimiento para llegar a los mejores resultados, con el apoyo de diferentes librerías y tecnologías que están preparadas para trabajos similares. Investigando sobre conceptos que ayudan a la detección rápida de objeto en las imágenes y determinando cuales son los más apropiados para la presente investigación se sacaron conclusiones sobre cuales posibles métodos son los más adecuados para la detección de objetos en imágenes.

3. Objetivos

3.1. Objetivo General

Desarrollar un software basado en matrices partiendo de los principios numéricos de Newton usados en el método YOLO, que intente encontrar los objetos que le sean indicados entregando la menor descripción posible del objeto.

3.2. Objetivos específicos

- Comparar el método de YOLO con el método de la conversión de colores para definir cual se adecúa más a lo que estamos trabajando.
- Aplicar conocimientos adquiridos de los métodos numéricos para optimizar el programa final, el cual será elaborado en Python.
- Realizar un estudio sobre los distintos tipos de redes neuronales, en especial sobre la red neuronal convolucional.

4. Marco teórico

El proyecto está basado en la localización de objetos dentro de imágenes. Las imágenes son realizadas mediante matrices, por lo cual es imprescindible usar librerías destinadas al control de las matrices.
Inicialmente estamos enfocados en dos de los muchos métodos que existen para realizar este tipo de proyectos. El primero sería el método YOLO y esta técnica utiliza un tipo de red neuronal convolucional llamada Darknet, el segundo método sería reconocer los objetos mediante su color pasando por un proceso de conversión de colores para al final destacar el objeto que estamos buscando. El problema de este último método es que necesitamos conocer de qué color es el objeto que estamos buscando, por lo cual trataremos de enfocarnos más en el primer método mencionado.

5. Introducción al machine learning

Para el desarrollo del reconocimiento de objetos en imágenes existe un concepto muy importante llamado machine learning. Dado a que es un concepto muy escuchado los últimos tiempos puede ser que ya sepas de que trata, sin embargo, vamos a abordar este tema para poder introducir nuestro tema central.

El machine learning o aprendizaje automático es una inteligencia artificial basada en varios algoritmos, todo esto con el fin de que los dispositivos aprendan mientras identifican patrones y realizan predicciones. Básicamente es la capacidad que tienen los computadores de realizar tareas específicas de manera autónoma.

El machine learning tiene demasiadas aplicaciones y es muy importante en la transformación digital que se está dando en la actualidad, algunas de sus aplicaciones son en las redes sociales, vehículos inteligentes, ciberseguridad, detección de rostros, entre otros. Nuestro objeto de estudio va muy de la mano con la detección de rostros dado que es una forma de reconocer “objetos” en una imagen.

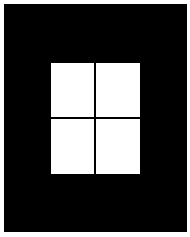
6. Reconocimiento de objetos en una imagen mediante su color

Existe un método utilizado para reconocer objetos en una imagen por medio de sus colores. Las imágenes son representadas por matrices donde podemos encontrar pixeles encargados de darle color por código RGB y es esto lo que hace posible la localización del objeto por medio de unos pasos donde se deben alterar los colores y conocer el código del color del objeto que estamos buscando.

Cuando importamos una imagen en un algoritmo, este la lee como si fuera una matriz conformada por pixeles. Estos pixeles son leídos como código RGB. Un ejemplo de una imagen en un algoritmo sería el siguiente:

0,0,0	0,0,0	0,0,0	0,0,0
0,0,0	255,255,255	255,255,255	0,0,0
0,0,0	255,255,255	255,255,255	0,0,0
0,0,0	0,0,0	0,0,0	0,0,0

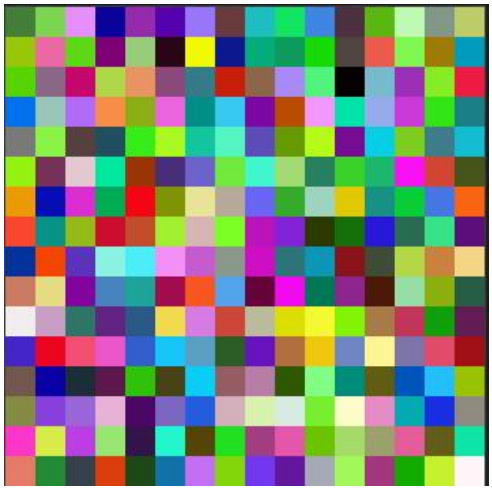
La matriz anterior representa una imagen de tamaño 4x4, utilizando 16 pixeles siendo 4 de estos blancos 12 negros, como se muestra en la siguiente imagen:



También se puede crear una imagen mediante un algoritmo usando las mismas bases anteriormente explicadas. A continuación, pueden observar un algoritmo elaborado en Python que se encarga de crear una imagen de tamaño 16x16 usando 256 pixeles creados de forma aleatoria en código RGB.

```python
import random
import numpy as np
import cv2

# creamos una matriz de 16*16 con todos los colores en blanco
img = np.zeros((16,16,3),np.uint8)

# recorremos cada uno de los elementos
for x in range(16):
    for y in range(16):

        # Cambiamos el color de cada uno de los pixeles de forma aleatoria
        img[x,y] = [random.randint(0,266),random.randint(0,266),random.randint(0,266)]

# guardamos la imagen png
cv2.imwrite("MyImage.png",img)
```

La imagen creada es la siguiente:



Cuando hablamos de RGB realmente nos referimos a una forma de representar el color mediante las siglas Red, Green, Blue. Esto sirve para representar diferentes colores con la mezcla de estos tres colores primarios. Los monitores de las computadoras representan los colores en código RGB. Para indicar la proporción de cada color, usamos unos intervalos de números que van de 0 hasta 255, siendo 0 la cantidad nula de este color y 255 la máxima. Para lograr un color se usa un valor para el campo R, otro para el campo G y otro para el campo B, de esta forma podemos decir que el rojo se obtiene con (255,0,0), el verde con (0,255,0) y el azul con (0,0,255).

A continuación, dejamos algunos ejemplos de colores.

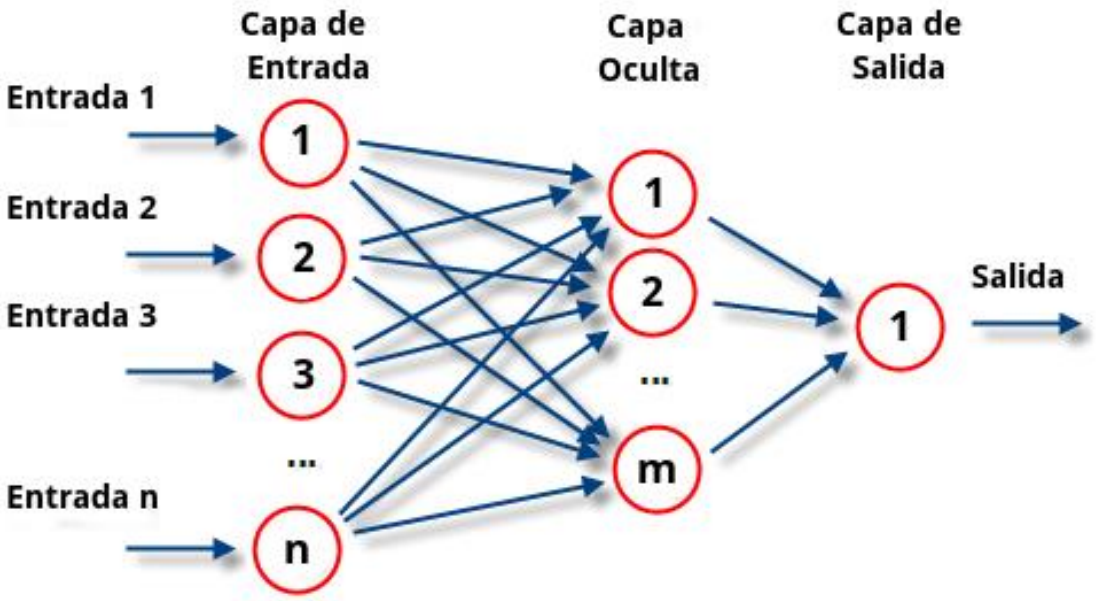
2	Color 0:		[Color 0]	255	255	255
3	Color 1:		[Color 1]	0	0	0
4	Color 2:			255	255	255
5	Color 3:		[Color 3]	255	0	0
6	Color 4:		[Color 4]	0	255	0
7	Color 5:		[Color 5]	0	0	255
8	Color 6:		[Color 6]	255	255	0
9	Color 7:		[Color 7]	255	0	255
10	Color 8:		[Color 8]	0	255	255
11	Color 9:		[Color 9]	128	0	0
12	Color 10:		[Color 10]	0	128	0
13	Color 11:		[Color 11]	0	0	128
14	Color 12:		[Color 12]	128	128	0
15	Color 13:		[Color 13]	128	0	128
16	Color 14:		[Color 14]	0	128	128
17	Color 15:		[Color 15]	192	192	192

Hemos realizado algunos algoritmos basándonos en tres teorías diferentes para localizar objetos, cabe recalcar que no todas funcionan al 100% porque siempre se maneja un porcentaje de error en estos tipos de algoritmos y en algunos puede ser mucho mayor el porcentaje de error que en otros.

A continuación, daremos una breve explicación de los principios de cada uno de los algoritmos implementados y aclararemos que fue necesario para poder implementar el algoritmo.

7. Algoritmos implementados:

7.1. Redes neuronales:



La figura de la imagen anterior es una red neuronal. Las redes neuronales están conformadas por capas, las cuales son capas de entrada, capas ocultas y capas de salida. Para el algoritmo que implementamos, los valores de entrada son los colores RGB, así que sólo usamos 3 entradas las cuales serían rojo, verde y azul.

Las redes neuronales son un sistema de aprendizaje automático que forma parte de la inteligencia artificial.

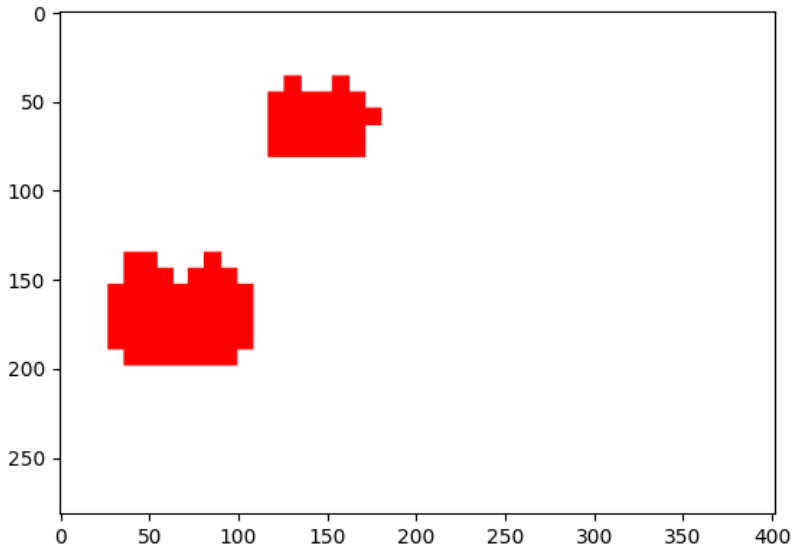
Para emplear este método en Python necesitamos las siguientes librerías:

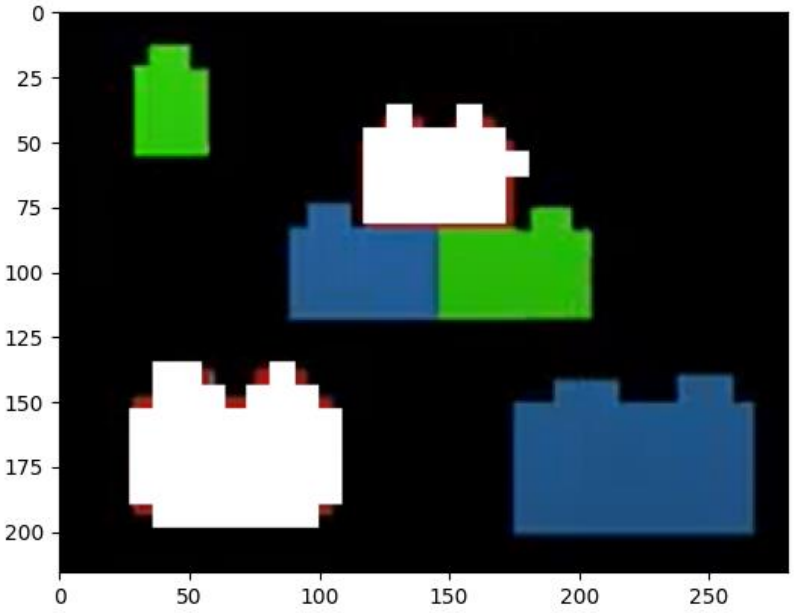
- Numpy
- Keras
- Matplotlib
- Cv2
- Sklearn

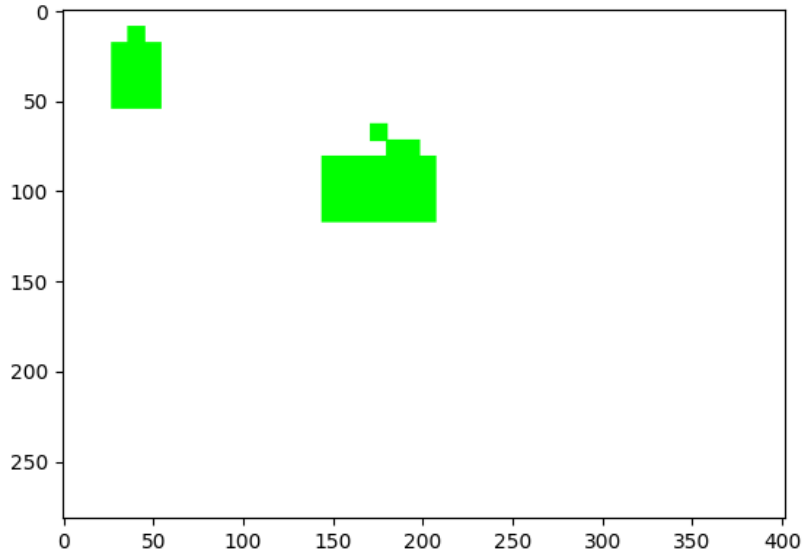
Funcionamiento del método:

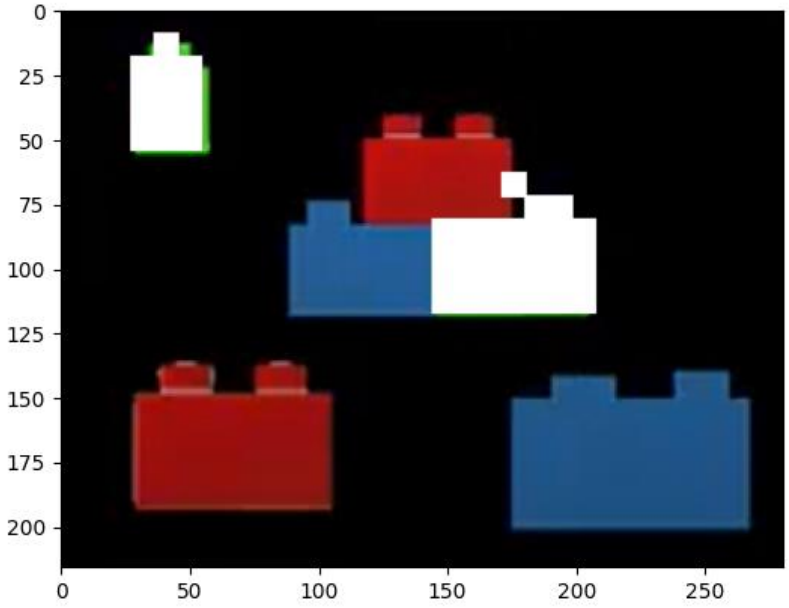
Este método consiste en entrenar una red neuronal para que se encargue de detectar los pixeles del color que estamos buscando, de manera que no se necesite tener el tono exacto del color, sino que la red neuronal los clasificará por rojo, verde o azul. Para entrenar la red neuronal debemos tener las dimensiones y la cantidad de pixeles de la imagen, así que, si queremos usar una imagen distinta, primero se deberán hacer algunos cambios en el código ya que las dimensiones y los pixeles serán diferentes y la red neuronal necesita los datos de la nueva imagen. Hay un problema de compatibilidad entre la librería cv2 y la librería matplotlib y es que, si subimos una imagen al algoritmo por cv2, al mostrarla por consola con numpy, se verá la imagen con los colores invertidos, esto es debido a que cv2 lee la imagen RGB como BGR y la sube al código como BGR, por lo cual es necesario cambiar el formato de la imagen de BGR a RGB. Luego de esto, se usa la función de Kmeans.predict para intentar separar los colores de la imagen en 4 clusters (rojo, verde, azul y negro) y luego de esto, para pasar los datos a la red neuronal, debemos convertir estos valores en términos de 0 y 1 puesto que usaremos la función sigmoide dentro de la red neuronal y esta sólo lee valores de 0 y 1. Luego de esto, la red neuronal se entrenará con el 70% de los pixeles de la imagen y se validará con el 30% de pixeles restantes, luego de esto, si la clasificación realizada por la red neuronal fue correcta, buscará el color que le indiquemos en la imagen y el objeto que allí encuentre, lo intentará borrar para dibujarlo en otra imagen en blanco llamada lienzo.

En las diferentes pruebas que hemos realizado con el método, hemos encontrado un porcentaje de error significativo ya que, en ocasiones la red neuronal no se entrena bien y confunde los colores.


0
50
100
150
200
250
0
50
100
150
200
250
300
350
400


0
25
50
75
100
125
150
175
200
0
50
100
150
200
250



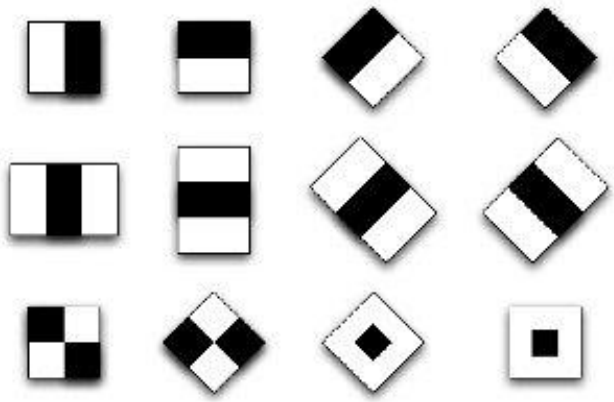


Estos son los resultados de encontrar los objetos rojos cuando la red se entrena bien, se puede observar como si hubiese dibujado los objetos en otra posición, pero en realidad lo que pasa es que el lienzo tiene más pixeles que la imagen original y se puede observar en la posición de los pixeles que se encuentran medidos. Los pedazos de color rojo que se observa que no fueron extraídos de la imagen son parte del porcentaje de error del algoritmo.

7.2. Reconocimiento por opencv 1:

La detección de objetos puede darse mediante clasificadores en cascada basados en funciones de Haar, esto se realiza con ayuda de machine learning o aprendizaje automático, para el código de reconocimiento de imágenes se necesita en principio una gran cantidad de imágenes de rostros (imágenes positivas) y luego una gran cantidad de imágenes sin rostros (imágenes negativas).

Para extraer rasgos del clasificador usamos las características de Haar mostradas en la siguiente imagen, donde cada característica es como un valor individual que se consigue con la resta de la suma de pixeles bajo rectángulo blanco y la suma de pixeles bajo rectángulo negro. Esto funciona como una red neuronal convolucional.



Clasificadores Haar

OpenCV es una librería que tiene incluido un entrenador y un detector, lo que ayuda al entrenamiento para la clasificación de objetos como lo son los coches, por ejemplo.

Para dar un ejemplo acorde a la realidad que vivimos, una aplicación de reconocimiento de objetos la vemos en la cámara de nuestro celular cuando esta detecta la cara de la persona. Esto funciona gracias a las redes neuronales convolucionales, estas redes son capaces de construir funciones a través de otras funciones menos complejas.

Para empezar la convolución es una operación matemática en la que una función aplica de alguna manera a otra función, en otras palabras, se puede decir que es como una mezcla de estas dos funciones. Las convoluciones son de mucha ayuda para la detección de estructuras de imágenes sencillas, este proceso lo hace sobre una serie de muchas capas. Como antes lo habíamos mencionado una imagen es una matriz de bytes, y la convolución recibe la imagen de esta manera.

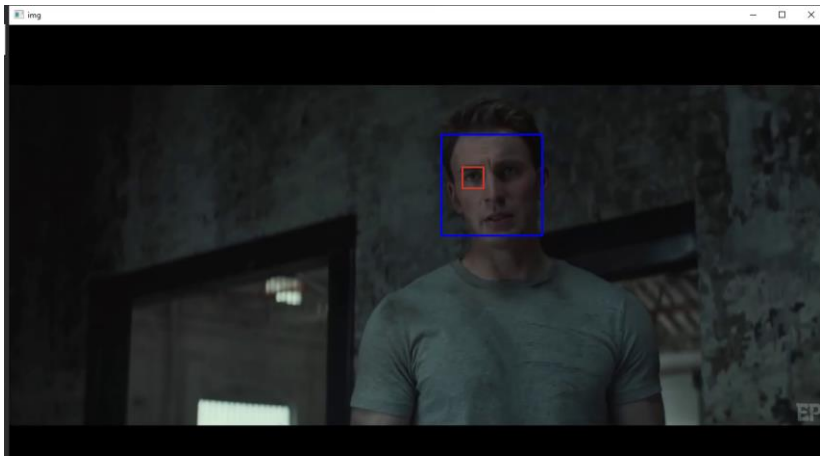
Primeramente, la imagen pasa a blanco y negro, y después se le aplica un filtro encardo de realizar un proceso con el fin de dar una nueva matriz de diferentes dimensiones

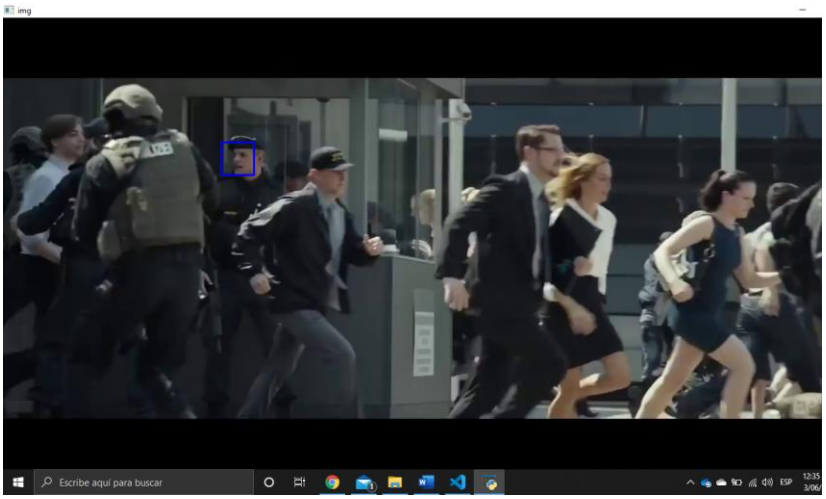
Imagen a blanco y negro

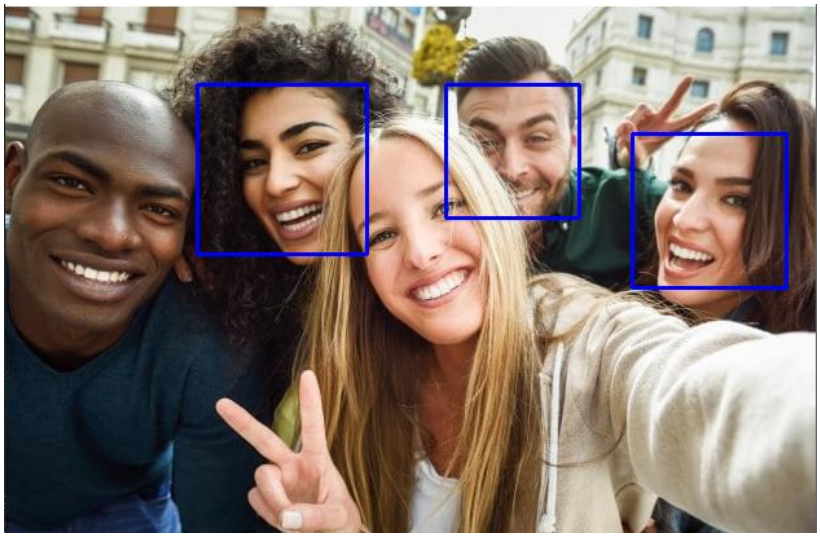
Imagen con el filtro

Funcionamiento del código:

El código es muy sencillo, gracias a la librería cv2, lo único que vamos a necesitar son algunos archivos haarcascade que son en formato xml, para este primer ejercicio vamos a trabajar con el haardcascade para reconocimiento de rostro y reconocimiento de ojos en cámara en vivo o en algún video.

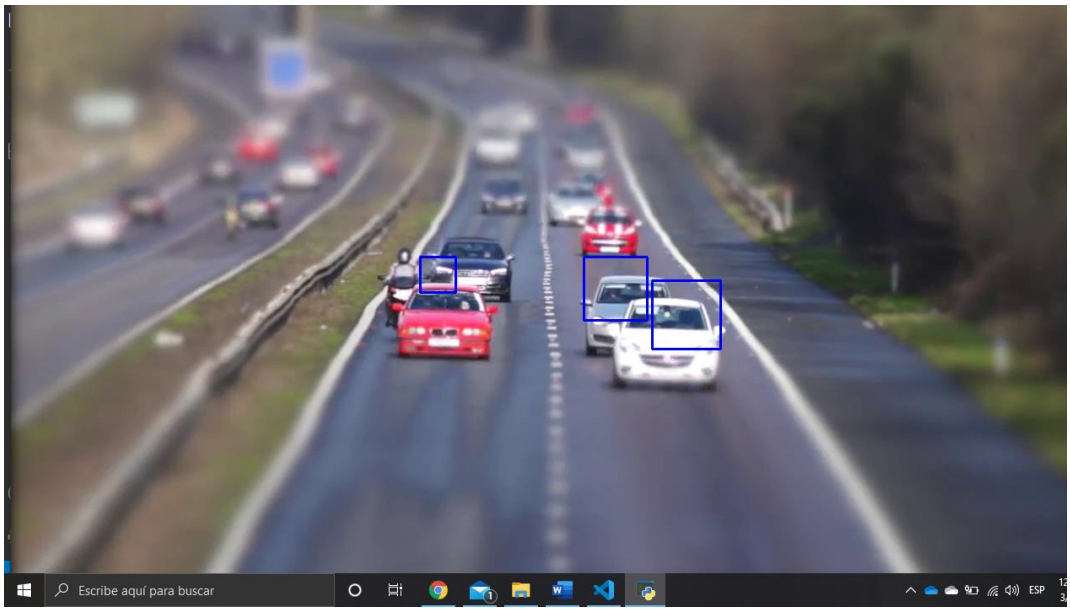

img


img
Escribe aquí para buscar
ESP



Las imágenes fueron captadas de un video, y podemos ver que no sa los resultado que esperaríamos, pero esto se da por el porcenaje de error que tiene este método, no es muy exacto y puede fallar, ya que es una alternativa rápida y una aproximación de la solución, más no es una solución definitiva.

Tambien se implementó un método de reconocimiento de carros, que al igual que el anterior también genera un porcentaje de error. Aquí una muestra de eso:

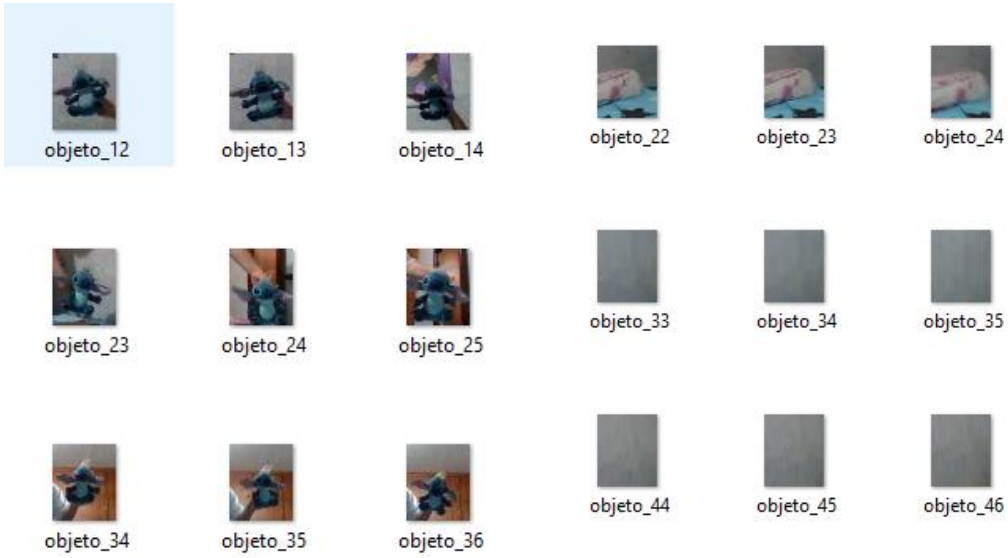


7.3. Reconocimiento por opencv 2:

Este método consiste en utilizar un programa llamado “cascade trainer gui” para entrenar un archivo .xml el cual tiene la información para detectar el objeto que se quiere identificar. El programa “cascade trainer gui” se utiliza para entrenar, probar y mejorar modelos de clasificadores en cascada. Los modelos de clasificadores en cascada tratan un enfoque basado en el aprendizaje automático en el que la función en cascada se entrena a partir de muchas imágenes positivas y negativas. Luego se utiliza para detectar objetos en otras imágenes. Se inicia llenando una carpeta con las imágenes del objeto que se desea reconocer, estas son las imágenes positivas para el programa, la carpeta de las imágenes positivas debe ser nombrada cómo “p”. Luego se procede a llenar una carpeta con las imágenes donde el objeto no está presente, estas son las imágenes negativas para el programa, se almacenan en una carpeta llamada “n”. Entre más imágenes se carguen en ambos casos mejores resultados se obtienen, se procede a utilizar el programa “cascade trainer gui” el cual es el que facilita el archivo .xml entrenado.

El código de reconocimiento consiste en leer el archivo .xml, el cual contiene la información que indica cuando está presente el objeto en la cámara.

Se procede a mostrar el contenido de la cámara en la pantalla y cuando en está aparece el objeto que se desea reconocer, entonces se muestra en un cuadro indicando que el objeto fue encontrado.


objeto_12
objeto_13
objeto_14
objeto_22
objeto_23
objeto_24
objeto_23
objeto_24
objeto_25
objeto_33
objeto_34
objeto_35
objeto_34
objeto_35
objeto_36
objeto_44
objeto_45
objeto_46

8. Conclusiones

- Los métodos de localización de objetos usualmente tienen un porcentaje de error significativo.
- Si vas a trabajar con matplotlib y cv2 para el procesamiento de imágenes, debes estar alternando el formato de la imagen ya que cv2 las carga como BGR y las lee como RGB y te puede generar errorres con matplotlib.
- Mientras más pixeles uses, mejor será el entrenamiento de las redes neuronales, aunque debes tener cuidado porque intentamos localizar objetos con 10 millones de pixeles y el algoritmo se demoraba más de 15 minutos, motivo por el cual fue descartado.
- El lenguaje de programación Python fue el usado para los métodos, gracias a su fácil adaptabilidad con la Inteligencia Artificial. Es considerado el mejor lenguaje para la IA
- Las redes neuronales son esenciales para trabajar este tipo de métodos, por lo que es importante conocer cómo funcionan dichas redes.
- OpenCV es una librería que nos facilita el proceso para la detección de objetos, en especial si es en tiempo real.
- OpenCV tiene aplicaciones a la robótica, medicina, seguridad, transporte, entre otros, en otras palabras, OpenCV se puede aplicar a muchas cosas del día a día.
- Las redes neuronales son aplicadas en muchas funciones como lo son la captura de objetos, reconocimiento de voces, reconocimiento facial, y la visión de un computador.

9. Bibliografía

- Aaron Nuñez Tv. (2021, 11 febrero). Detección de objetos | Procesamiento de Imágenes con Python | Keras - KMeans - Bordes - Colab. Recuperado de https://www.youtube.com/watch?v=51fW04hNzhM
- El modelo de redes neuronales. (s. f.). Recuperado de https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model
- Código de Python - Crear un imagen con cada pixel de un color. (2019, 28 febrero). Recuperado de https://www.lawebdelprogramador.com/codigo/Python/5175-Crear-un-imagen-con-cada-pixel-de-un-color.html
- Cascade Trainer GUI. (2020, 30 mayo). Recuperado de https://amin-ahmadi.com/cascade-trainer-gui/
- OpenCV: Cascade Classifier Training. (s. f.). Recuperado 2 de junio de 2021, de https://docs.opencv.org/4.3.0/dc/d88/tutorial_traincascade.html
- E. (s. f.). XML Ã‚Â¿QuÃƒÂ© es? | Manual de XML. Recuperado 2 de junio de 2021, de https://www.mundolinux.info/que-es-xml.htm
- Fernandez, R. (2019, 30 agosto). Detección de rostros, caras y ojos con Haar Cascad. Recuperado de https://unipython.com/deteccion-rostros-caras-ojos-haar-cascad/#:%7E:text=Se%20trata%20de%20un%20enfoque,detecci%C3%B3n%20de%20rostros%20o%20caras.
- (2019, 14 febrero). Qué Es El «Machine Learning». Recuperado 2 de junio de 2021, de https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico